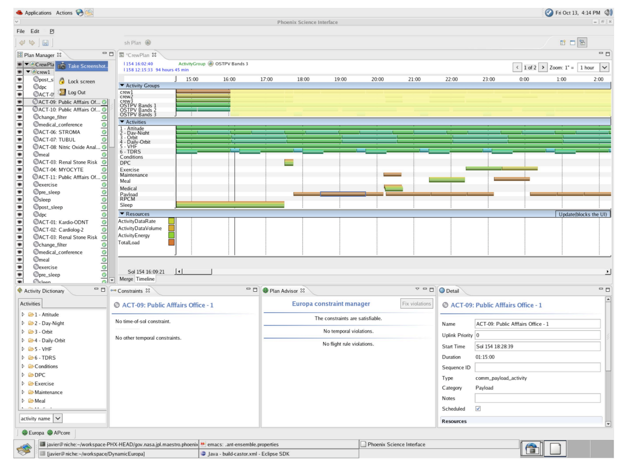
Verification of Model-based Software Systems for Space Applications

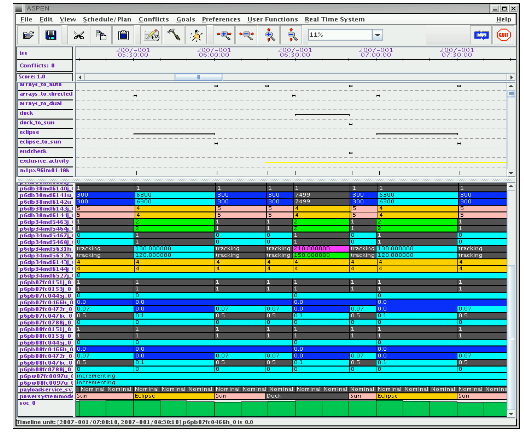
Automatic generation of tests for planner

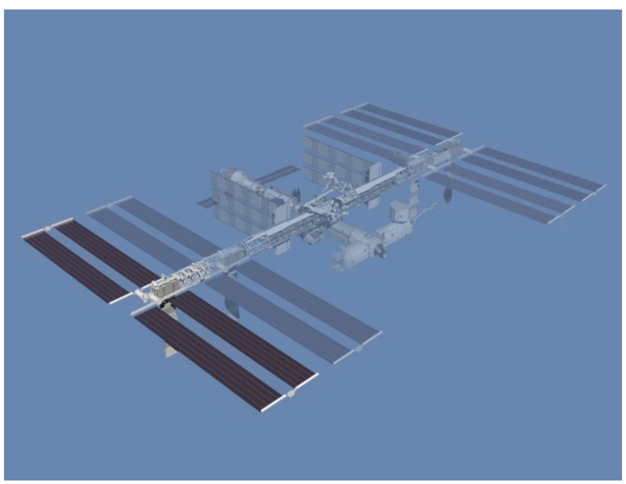
- Goal:
  - automatically generate test cases for planners so that we can test planning domains against flight rules
- Contributors:
  - Franco Raimondi (University College of London, UK)
  - Charles Pecheur (Université Catholique de Louvain, Belgium)
  - Guillaume Brat (USRA-RIACS, USA)
- Funding:
  - NASA Exploration Technology Development Program
  - Autonomy for Operations project



The EUROPA planning framework

- Class library and tool set for building planners (and/or schedulers) within a Constraint-based Temporal Planning paradigm.
  - explicit notion of time
  - deep commitment to a constraint-based formulation of planning problems





- Applications
  - MAPGEN: used on MER mission
  - SACE: used for ISS solar arrays

New Domain Description Language

- NDDL roles
  - Describe domain types and domain rules
    - Includes activity types, subgoal rules, resources, limits, constraints,…
  - Describe instantiations of activities, resources, constraints,…

```
class Path {
  string name;
  Loc from;
  Loc to;
  float cost;
  Path(string name,
       Loc from,
       Loc to,
       float cost) {
    name = name;
    from = from;
    to = to;
    cost = cost;
  }
}
```

```
class Rover {
  predicate At {
    Loc location;
  }
  predicate Going {
    Loc from;
    Loc to;
    neq(from, to);
  }
}
```

```
Rover::Going{
  met_by(object.At at_before);
  eq(at_before.location, from);
  meets(object.At at_after);
  eq(at_after.location, to);

  Path p : {
    eq(p.from, from);
    eq(p.to, to);
  };

  starts(Battery.change tx);
  eq(tx.quantity, p.cost);
}
```

Rover Flight Rules in LTL

FR1. The Rover Battery State of charge cannot go below X

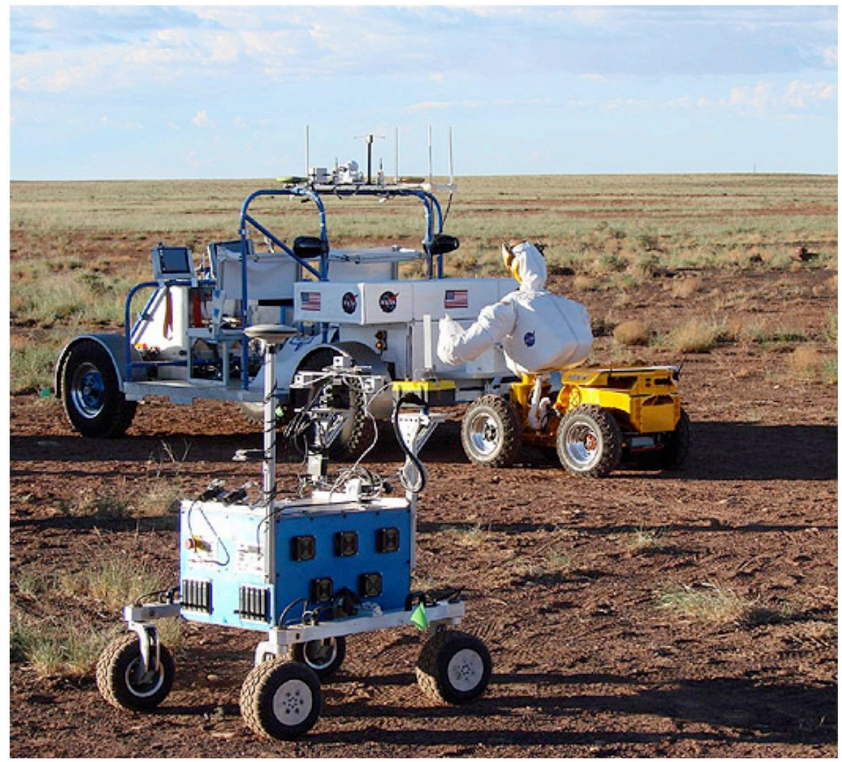
G (battery::state(level) and level ≥ X)

FR2. All Instruments must be stowed when moving

**G (SPIRIT.navigator::moving(*,*)
implies
SPIRIT.instrument::stowed())**

FR3. The Rover may only navigate along designated paths

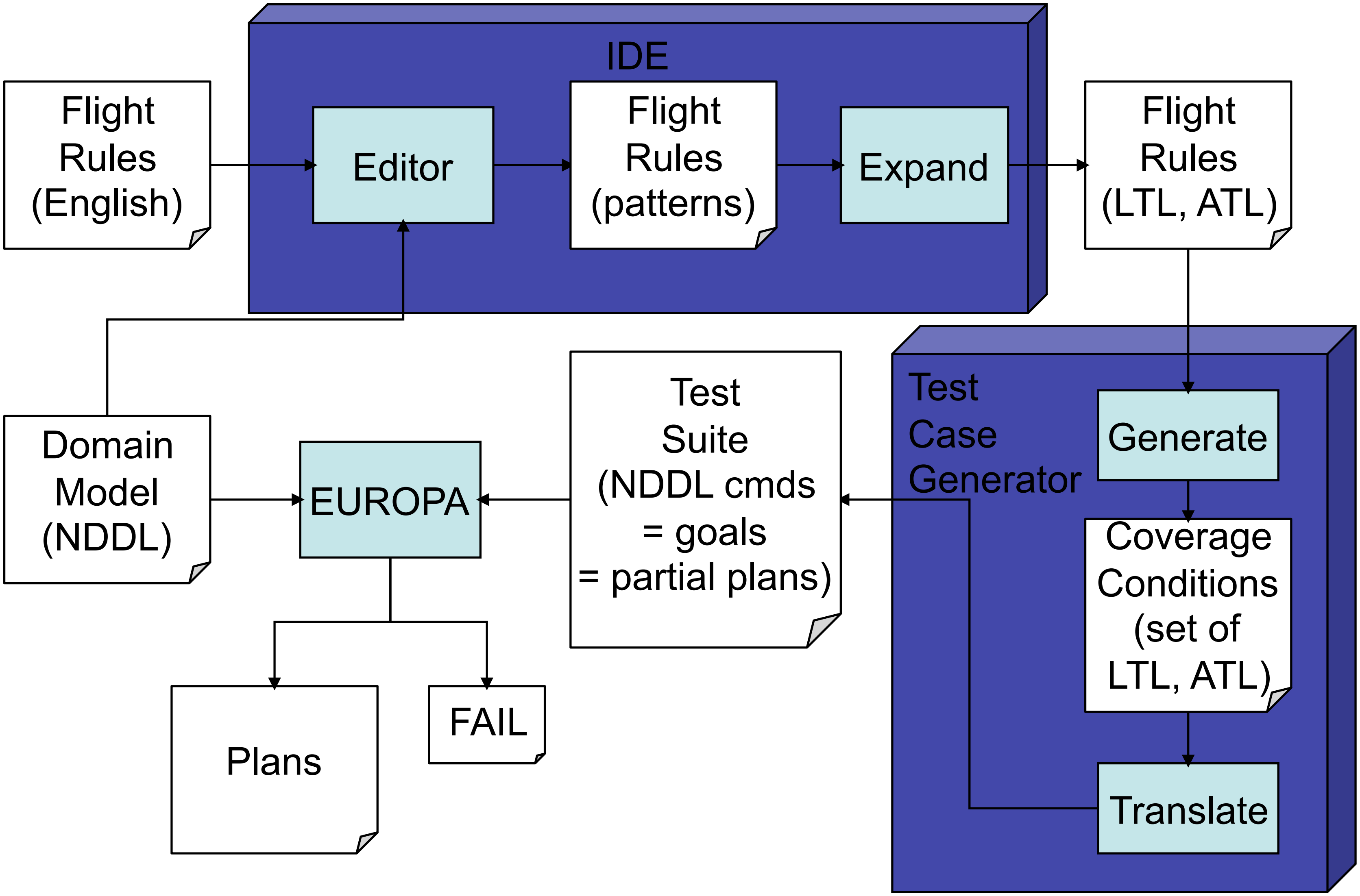
**G(SPIRIT.navigator::going(from,to)
implies
(from=FROM1 and to=TO1) or (...)
or ...)**



Process

- Modeling flight rules in appropriate language
  - We started with LTL (linear temporal logic), but are considering others
  - Flight rules will likely be expressed in plain English
- Generate coverage conditions that cover flight rules according to “unique cause” criterion
  - “Unique cause” is an extension of the commonly used MC/DC coverage criterion mandated by the FAA
- Generate test case in the form of Europa goals (or partial plans) using the coverage conditions

Test case generation for NDDL



Unique First Cause Coverage

- Idea: extend MC/DC testing)mandated by FAA for avionics) to requirement-based testing
- A test suite achieves UFC coverage of a set of requirements expressed as temporal formulae, if:
  1. Every basic condition in any formula has taken on all possible outcomes at least once
  2. Each basic condition has been shown to affect the formula's outcome as the unique first cause
- A condition a is the unique first cause (UFC) for φ along a path π if, in the first state along π in which φ is satisfied, it is satisfied because of a.